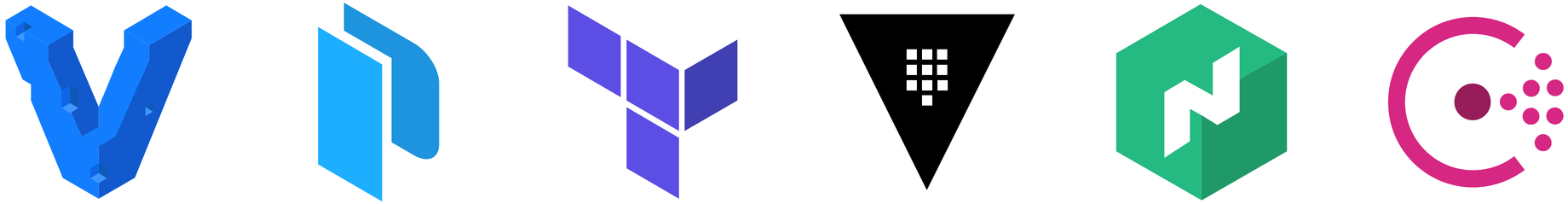
400: Terraform tips and tricks

Tom Harvey @tombuildsstuff

Terraform Engineer, AzureRM Provider

Copyright © 2018 HashiCorp



Providers

Multiple instances of a Provider, chaining providers together

Copyright © 2018 HashiCorp

Multiple instances of a single provider

Configuring multiple instances of a single provider

```
provider "azurerm" {
  version         = "=1.8.0"
  subscription_id = "00000000-0000-0000-0000-000000000000"
  alias           = "ci"
}

provider "azurerm" {
  version         = "=1.8.0"
  subscription_id = "11111111-1111-1111-1111-111111111111"
  alias           = "preprod"
}
```

Copyright © 2018 HashiCorp

Multiple instances of a single provider

Using multiple instances of a single provider

```
resource "azurerm_resource_group" "first" {
  name     = "ci-resources"
  provider = "azurerm.ci"
}

resource "azurerm_resource_group" "second" {
  name     = "preprod-resources"
  provider = "azurerm.preprod"
}
```

Copyright © 2018 HashiCorp

Chaining together multiple providers

Provision AKS and then provision some Pods

```
provider "azurerm" { ... }

module "kubernetes-cluster" { /* provisions an AKS cluster */ }

provider "kubernetes" {
  version                 = "=1.1.0"
  host                    = "${module.kubernetes-cluster.host}"
  username                = "${module.kubernetes-cluster.username}"
  password                = "${module.kubernetes-cluster.password}"
  client_certificate      = "${base64decode(module.kubernetes-
cluster.client_certificate)}"
  client_key              = "${base64decode(module.kubernetes-
cluster.client_key)}"
  cluster_ca_certificate = "${base64decode(module.kubernetes-
cluster.cluster_ca_certificate)}"
  load_config_file        = false
}
```

Example module for AKS: github.com/tombuildsstuff/microsoft-build2018

Copyright © 2018 HashiCorp

Provisioners

Running commands on Virtual Machines

Copyright © 2018 HashiCorp

Provisioners

Running commands on resource creation

- Provisioners allow you to run commands when any resource is created (e.g. a VM / Function resource)

- Provisioners for:
  - Uploading Files (file)
  - Running commands on a remote machine (remote-exec)
  - Running commands on your local machine (local-exec)
  - Running Chef on a remote machine (chef)
  - etc

Copyright © 2018 HashiCorp

Provisioners

Step 1: Defining the Connection

```
resource "azurerm_virtual_machine" "main" {
  # ...

  connection {
    type     = "ssh"
    host     = "${azurerm_public_ip.main.ip_address}"
    user     = "${var.username}"
    password = "${var.password}"
  }
}
```

Copyright © 2018 HashiCorp

Provisioners

Step 2: Defining the Provisioner

```
resource "azurerm_virtual_machine" "main" {
  # ...

  connection { ... }

  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install -y golang",
    ]
  }
}
```

Copyright © 2018 HashiCorp

Provisioners

Step 3: Terraform Apply

```
$ terraform apply

azurerm_virtual_machine.main: Provisioning with 'remote-exec'...
azurerm_virtual_machine.main (remote-exec): Connecting to remote
host via SSH...
azurerm_virtual_machine.main (remote-exec):   Host: 40.115.38.54
azurerm_virtual_machine.main (remote-exec):   User: tharvey
azurerm_virtual_machine.main (remote-exec):   Password: true
azurerm_virtual_machine.main (remote-exec):   Private key: false
azurerm_virtual_machine.main (remote-exec):   SSH Agent: true
azurerm_virtual_machine.main (remote-exec):   Checking Host Key:
false
azurerm_virtual_machine.main (remote-exec): Connected!
azurerm_virtual_machine.main (remote-exec): 0% [Working]
azurerm_virtual_machine.main (remote-exec): Get:1 http://
security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
```

Copyright © 2018 HashiCorp

Demo

Installing Go via a Provisioner

Copyright © 2018 HashiCorp

Provisioners

Summary

- Providers generally set the `connection` block for you
- Provisioners can be run from any resource
- The `null_resource` can be particularly useful for running provisioners
- You can use multiple Provisioners together as needed (e.g. copy the files then run this script)

Copyright © 2018 HashiCorp

Terraform Modules

Registry, scope of a module

Copyright © 2018 HashiCorp

Terraform Modules

Module sources

- Terraform Modules can be sourced from:
  - Public Terraform Registry
  - Any Git repository
  - Private Terraform Registry

Copyright © 2018 HashiCorp

Terraform Modules

Module sources

- Terraform Modules can be sourced from:
  - Public Terraform Registry
  - Any Git repository
  - Private Terraform Registry
- Terraform Modules can be versioned via Git Tags

Copyright © 2018 HashiCorp

Terraform Module Registry

registry.terraform.io/modules/hashicorp/nomad/azurerm/0.0.1

Terraform | Module Registry

Search | Browse | Publish | Sign-in

nomad

AZURERM

Version 0.0.1

A Terraform Module for how to run Nomad on AzureRM using Terraform and Packer

Published October 13, 2017 by hashicorp

Module managed by gruntwork-team

Total provisions: < 100

Source: github.com/hashicorp/terraform-azurerm-nomad (report an issue)

Submodules

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run `terraform init`:

```
module "nomad" {
  source  = "hashicorp/nomad/azurerm"
  version = "0.0.1"

  # insert the 8 required variables here
}
```

Readme | Inputs (18) | Outputs (4) | Dependency (1) | Resources (10)

Nomad Azure Module

This repo contains a Module for how to deploy a Nomad cluster on Azure using Terraform. Nomad is a distributed, highly-available data-center aware scheduler. A Nomad cluster typically includes a small number of server nodes, which are responsible for being part of the concensus protocol, and a larger number of client nodes, which are used for running jobs:

Copyright © 2018 HashiCorp

Building Modules

- Terraform Modules should be specific not generic
- This allows modules to be composed together as needed - for example:
  - Virtual Machine
  - Virtual Machine with 2 x Data Disks

Copyright © 2018 HashiCorp

Building Modules

- Given two Virtual Networks (either in the same subscription, or cross-subscription) - how's best to create a Terraform Module to create a Peering in both directions?

- How's best to structure this?
  - One module which takes both Providers and configures the connection?
  - One module called twice which configures the connection in both ends?

Copyright © 2018 HashiCorp

Suggestions for Terraform Modules

- Virtual Machine Extensions
  - Joining an Active Directory domain
  - DataDog Agent
  - Network Watcher Agent
  - VSTS Agent
  - PowerShell DSC
  - Octopus Deploy Tentacle

Copyright © 2018 HashiCorp

Suggestions for Terraform Modules

- Networking:
  - Network Peerings
  - Application Gateway configurations
  - Load Balancers
- PaaS:
  - Azure Functions
  - App Services

Copyright © 2018 HashiCorp

The Terraform Statefile

Understanding the Statefile, Backends and Refactoring

Copyright © 2018 HashiCorp

Understanding the Statefile

What's it for?

- Terraform's Statefile is used to keep track of what Terraform's provisioned
- The Statefile stores a cache of all resources Terraform's touched, which allows us to provide the Diff.

Copyright © 2018 HashiCorp

Understanding the Statefile

Storing the Statefile

- By default this is stored locally, however we recommend storing this in a Backend
- Backends provide both State storage and allow for Locking (e.g. so only Terraform Apply can happen at a time)

Copyright © 2018 HashiCorp

Backends

What's in a Backend?

- All Backends provide State Storage
- Backend-dependent:
  - Locking
  - Encryption at rest
  - Versioning

Copyright © 2018 HashiCorp

Backends

Which backends are available?

- Backends are available for:
  - AWS S3
  - Azure Blob Storage
  - Consul
  - Google Cloud Storage
  - (and more)

Copyright © 2018 HashiCorp

Backend Configuration

Configuring the AzureRM Backend

```
terraform {
  backend "azurerm" {
    storage_account_name = "abcd1234"
    container_name       = "tfstate"
    key                  = "prod.terraform.tfstate"
  }
}
```

Copyright © 2018 HashiCorp

Sentinel

Policy as Code

Copyright © 2018 HashiCorp

Disclaimer

This requires Terraform Enterprise

Copyright © 2018 HashiCorp

Sentinel

Policy as Code

- Policy as Code framework
- An SDK allows integration with internal systems
- Three-Levels for Policy Enforcement:
  - Hard-Mandatory (must pass)
  - Soft-Mandatory (overridable)
  - Advisory (logging only)

Copyright © 2018 HashiCorp

Sentinel Example

Ensuring all resource groups have at least one tag

```
import "tfplan"

main = rule {
  all tfplan.resources.azurerm_resource_group as _, group {
    all group as _, g {
      (length(g.applied.tags) else 0) > 0
    }
  }
}
```

Copyright © 2018 HashiCorp

Sentinel Example

Given this Terraform Code

```
resource "azurerm_resource_group" "test" {
  name     = "tom-dev2"
  location = "West Europe"
}
```

Copyright © 2018 HashiCorp

tom-tfe.hashicorptest.com/app/exampleinc/azurerm-talk-mstvp/current

PLAN

Queued 14 minutes ago
Started 14 minutes ago

Executed and saved successfully
13 minutes ago

View Plan

State versions output with this plan

No state versions output

POLICY CHECK

Queued 13 minutes ago

Organization policies hard failed
a few seconds ago

View Check

Leave a comment

 © 2018 HashiCorp, Inc.

tom-tfe.hashicorptest.com/app/exampleinc/azurerm-talk-mstvp/runs/run-KomVw5GuZqV

Queued a few seconds ago
Started a few seconds ago

Executed and saved successfully
a few seconds ago

View Plan

State versions output with this plan

No state versions output

POLICY CHECK

Queued a few seconds ago

Organization policies soft failed
a few seconds ago

View Check

Leave a comment

Override & Continue

 © 2018 HashiCorp, Inc.

tom-tfe.hashicorptest.com/app/exampleinc/azurerm-talk-mstvp/current

Queued a few seconds ago
Started a few seconds ago

Executed and saved successfully
a few seconds ago

View Plan

State versions output with this plan

No state versions output

POLICY CHECK

Queued a few seconds ago

Organization policies failed (advisory)
a few seconds ago

View Check

Leave a comment

Confirm & Apply Discard Plan

 © 2018 HashiCorp, Inc.

HashiConf '18

October 22-24, 2018 | San Francisco

Thank you.

HashiCorp

www.hashicorp.com hello@hashicorp.com